WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Friday, August 20, 2004

Hide?	Set Name	Query	Hit Count
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ		
☐	L8	(populate or populating) near8 (set or block) near8 array near8 (immage or image or copy)	3
☐	L7	(populate or populating) near8 (set or block) near8 array	29
☐	L6	20010522	6
☐	L5	(computer adj3 (program or programming)) near8 (pass or passing) near8 (parameter or argument) near8 (block or set)	7
☐	L4	l3 and ((copy or immage) near5 array)	37
☐	L3	20010522	784
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR		
☐	L2	((pass or passing) near8 (parameter or argument) near8 (program or programming))	1004
☐	L1	((pass or passing) near8 (parameter or argument) near8 (program or programming))	1004

END OF SEARCH HISTORY

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

☐ Generate Collection

L4: Entry 10 of 37 File: USPT Feb 10, 2004

DOCUMENT-IDENTIFIER: US 6691301 B2
TITLE: System, method and article of manufacture for signal constructs in a programming language capable of programming hardware architectures

Application Filing Date (1):
20010129

Detailed Description Text (758):
Starting up process-hogging models is a little more involved. For a light-weight model to instantiate a process-hogging model, the light-weight model may know the name of a simulator specific launcher dll. This name is passed to Cosim HO which gives the launcher dll details of how IPC is to be achieved. The launcher dll then loads up the simulator which may at some point load up the simulator specific cosim plugin, which loads up a generic cosim dll. The simulator specific launcher and cosim plugins may cooperate is passing the IPC connection information from Cosim HO to the generic cosim dll. Once this has been achieved communication between the two processes can take place. The techniques described here avoid the simulator specific plugins needing to know how IPC takes place, and avoids the cosimulation program needing to know how to start up and pass parameters to every different kind of simulator.

Detailed Description Text (1368):
static gives a variable static storage (its values are kept at all times). This ensures that the value of a variable is preserved across function calls. It also affects the scope of a variable or a function. Static functions and static variables declared outside functions can only be used in the file in which they appear. static variables declared within an inline function or an array of functions can only be used in the copy of the function in which they appear. static variables are the only local variables (excluding consts) that can be initialized.

Detailed Description Text (1568):
Functions are similar to functions in ISO-C. Handel-C has been extended to provide arrays of functions and inline functions. Arrays of functions provide multiple copies of a function. One can select which copy is used at any time. Inline functions are similar to macros in that they are expanded wherever they are used. Functions take arguments and return values. A function that does not return a value is of type void. The default return type is int undefined.

Detailed Description Text (1581):
A function array allows one to run different copies of the function in parallel. Without this construct, the only safe way to run a function in parallel with itself would be to explicitly declare two functions with different names. This would not be so neat and intuitive.

Previous Doc Next Doc Go to Doc#

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L4: Entry 16 of 37 File: USPT Sep 7, 1999

DOCUMENT-IDENTIFIER: US 5949976 A
TITLE: Computer performance monitoring and graphing tool

Application Filing Date (1):
19960930

Detailed Description Text (70):
In step 812, parser script 502 calls conversion program 504, passing two arguments: the temporary filename and the destination GIF file, triggering the execution of conversion program 504. After parser script 502 triggers conversion program 504, operation of flowchart 800 is complete, as indicated in step 814.

Detailed Description Text (72):
In step 904, conversion program 504 is initiated by the call from parser script 502. In step 906, conversion program 504 copies data log 410 into an array. Data log 410 is stored as a temporary file.

Previous Doc Next Doc Go to Doc#

Previous Doc Next Doc Go to Doc#
First Hit

Generate Collection

L8: Entry 2 of 3 File: PGPB Oct 31, 2002

DOCUMENT-IDENTIFIER: US 20020159632 A1
TITLE: Graphic image re-encoding and distribution system and method

Detail Description Paragraph:
[0106] The coding type identifier is typically read (380) by a main decoder procedure, which then calls the remapping and run length decoder on the basis of the coding type identifier in the block image file. The remapping and run length decoder reads (inputs) and stores the remapping table 270 (FIG. 6C) stored in the block image file (382), and then decodes the run length data in the block image file and populates a block image array with the decoded values (384). More specifically, each successive (run length, pixel value) tuple in the block image file is read, and the specified pixel value is written into the next "run length" pixel positions of the block image array. The order of the pixels in the block image array depends on the scan pattern used by the encoding procedure that produced the block image file, which in turn is represented by either the coding type at the beginning of the file, or the coding type and a sub-type value, as shown in FIG. 12B. After the last tuple in the block image file has been decoded, the pixel values are remapped into palette index values (386) using the remapping table read in step 382.

Previous Doc Next Doc Go to Doc#